This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Currently Amended) A method for allowing users to define new aggregates in a database system, comprising:

receiving code that instantiates an object of a class defining the structure of a user-defined aggregate and methods that can be invoked on instances of the user-defined aggregate; and

enforcing a contract against the class so that the code, when executed at runtime, satisfies requirements of the contract to ensure desired execution thereof, the contract requiring that the class comprise a first method that is invoked to initialize the computation of an instance of the user-defined aggregate, a second method that is invoked to accumulate a plurality of values to be aggregated, and a third method that is invoked to compute a final result of the instance of the user-defined aggregate, the contract further comprising a requirement that the class specify one of a plurality of different formats which describes how instances of the user-defined aggregate are to be persisted in a database store, wherein the plurality of different formats which describe how instances of the user-defined aggregate are to be persisted in the database store comprises:

a first format in which an instance of the user-defined aggregate is automatically serialized in accordance with a native format of the database system; and

a second format in which an instance of the user-defined aggregate is serialized in a manner defined by the class.

2. (Canceled)

3. (Previously Presented) The method of claim 1, wherein the contract further comprises a requirement that the class comprise a fourth method that is invoked to merge an instance of the user-defined aggregate with another partial aggregation.

4. (Original) The method of claim 1, wherein the contract further comprises a requirement that the class comprise a public constructor having no arguments.

5. (Original) The method of claim 1, further comprising storing metadata about the user-defined aggregate for subsequent use by the database system in creating instances of the user-defined aggregate.

6. (Canceled)

7. (Currently Amended) The method of claim [[6]] 1, wherein the plurality of different formats which describe how instances of the user-defined aggregate are to be persisted in the database store further comprises a third format in which an instance of the user-defined aggregate is serialized in accordance with a method provided by a managed code programming model.

8. (Original) The method of claim 1, wherein the code that implements the class comprises managed code.

9. (Previously Presented) The method of claim 1, further comprising:

receiving a query that requires evaluation of the user-defined aggregate over a group of elements;

instantiating an instance of the class that defines the structure of the user-defined aggregate;

invoking said first method to initializing the instance of the user-defined aggregate;

for each element of the group, invoking said second method on the instance of the user-defined aggregate to accumulate each element; and

invoking said fourth method to return a value of the user-defined aggregate.

10. (Original) The method of claim 9, further comprising deserializing the instance of the user-defined aggregate prior to invoking said second method for an element of the group and then serializing the instance after invoking said second method for the element.

11. (Original) The method of claim 9, further comprising:

deserializing the instance of the user-defined aggregate prior to invoking said second method for a first element of the group;

caching the deserialized instance of the user-defined aggregate;

for each element of the group, invoking said second method on the cached instance of the user-defined type; and, thereafter,

serializing the cached instance of the user-defined type.

12. (Previously presented) The method of claim 1, wherein the class defining the user-defined aggregate is annotated to specify one of:

an attribute that indicates whether the user-defined aggregate is invariant to duplicates;

an attribute that indicates whether the user-defined aggregate is invariant to NULL values;

an attribute that indicates whether the user-defined aggregate is invariant to order; and

an attribute that indicates whether the user-defined aggregate returns NULL if the group on which the user-defined aggregate is to be computed is empty.

13. (Previously Presented) The method of claim 12, further comprising determining a method of computation of an instance of the user-defined aggregate based at least in part upon a value of said one of said attributes.

14. (Currently Amended) A database system that allows users to define new aggregates, comprising:

a runtime that provides code execution within the database system; and

a database server that receives code that instantiates an object of a class defining the structure of a user-defined aggregate and methods that can be invoked on instances of the user-defined aggregate and that enforces a contract against the class so that the code, when executed at runtime, satisfies requirements of the contract to ensure desired execution thereof, the contract requiring that the class comprise a first method that is invoked to initialize the computation of an instance of the user-defined aggregate, a second method that is invoked to accumulate a plurality of values to be aggregated, and a third method that is invoked to

compute a final result of the instance of the user-defined aggregate, the contract further comprising a requirement that the class specify one of a plurality of different formats which describes how instances of the user-defined aggregate are to be persisted in a database store, wherein the plurality of different formats which describe how instances of the user-defined aggregate are to be persisted in the database store comprises:

a first format in which an instance of the user-defined aggregate is automatically serialized in accordance with a native format of the database system; and

a second format in which an instance of the user-defined aggregate is serialized in a manner defined by the class.

15. (Canceled)

16. (Original) The system of claim 14, wherein the contract further comprises a requirement that the class comprise a fourth method that is invoked to merge an instance of the user-defined aggregate with another partial aggregation.

17. (Original) The system of claim 14, wherein the contract further comprises a requirement that the class comprise a public constructor having no arguments.

18. (Original) The system of claim 14, wherein the database server stores metadata about the user-defined aggregate for subsequent use by the database system in creating instances of the user-defined aggregate.

19. (Canceled)

20. (Currently Amended) The system of claim [[19]] 14, wherein the plurality of different formats which describe how instances of the user-defined aggregate are to be persisted in the database store further comprises a third format in which an instance of the user-defined aggregate is serialized in accordance with a method provided by a managed code programming model.

21. (Original) The system of claim 14, wherein the code that implements the class comprises managed code.

22. (Previously Presented) The system of claim 14, wherein the database server (i) receives a query that requires evaluation of the user-defined aggregate over a group of elements; (ii) instantiates an instance of the class that defines the user-defined aggregate; (iii) invoking said first method to initializing the instance of the user-defined aggregate; (iv) for each element of the group, invokes said second method on the instance of the user-defined aggregate to accumulate each element; and (iv) invokes said fourth method to return a value of the user-defined aggregate.

23. (Original) The system of claim 22, wherein the database server deserializes the instance of the user-defined aggregate prior to invoking said second method for an element of the group and then serializes the instance after invoking said second method for the element.

24. (Original) The system of claim 22, wherein the database server (i) deserializes the instance of the user-defined aggregate prior to invoking said second method for a first element of the group; (ii) caches the deserialized instance of the user-defined aggregate; (iii) for each element of the group, invokes said second method on the cached instance of the user-defined type; and (iv) thereafter, serializes the cached instance of the user-defined type.

25. (Previously presented) The system of claim 14, wherein the class defining the user-defined aggregate is annotated to specify one of:

an attribute that indicates whether the user-defined aggregate is invariant to duplicates;

an attribute that indicates whether the user-defined aggregate is invariant to NULL values;

an attribute that indicates whether the user-defined aggregate is invariant to order; and

an attribute that indicates whether the user-defined aggregate returns NULL if the group on which the user-defined aggregate is to be computed is empty.

26. (Previously Presented) The system of claim 25, further comprising a query processor that determines a method of computation of an instance of the user-defined aggregate based at least in part upon a value of said one of said attributes.

27. (Original) A computer-readable medium having program code stored thereon for allowing users to define new aggregates in a database system, the program code, when executed by a computer, causing the computer to perform the method recited in claim 1.